DEPARMENT OF INFORMATION AND COMMUNICATION ENGINEERING

TP 11

VUEJS, NODEJS, CRUD APIS, MONGOOSE

INTERNET PROGRAMMING

Submitted to: lecturer HOK TIN

e20190146
Chhorn Kakada
GIC-A

Tuesday. 13. June. 2023. 11:11PM

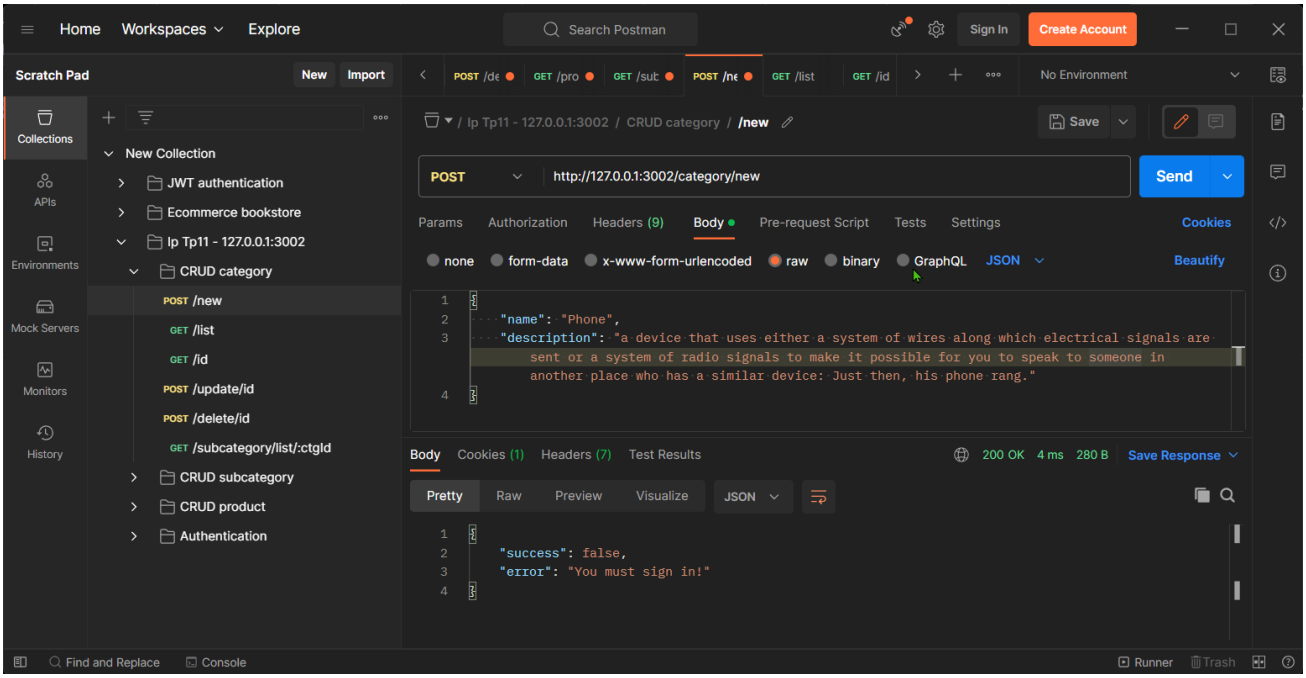
Github Account:
https://github.com/ChhornKakada/IntenetProgramming/tree/tp11

Noted: in this tp11, I only done with API want do it with **NestJs**.

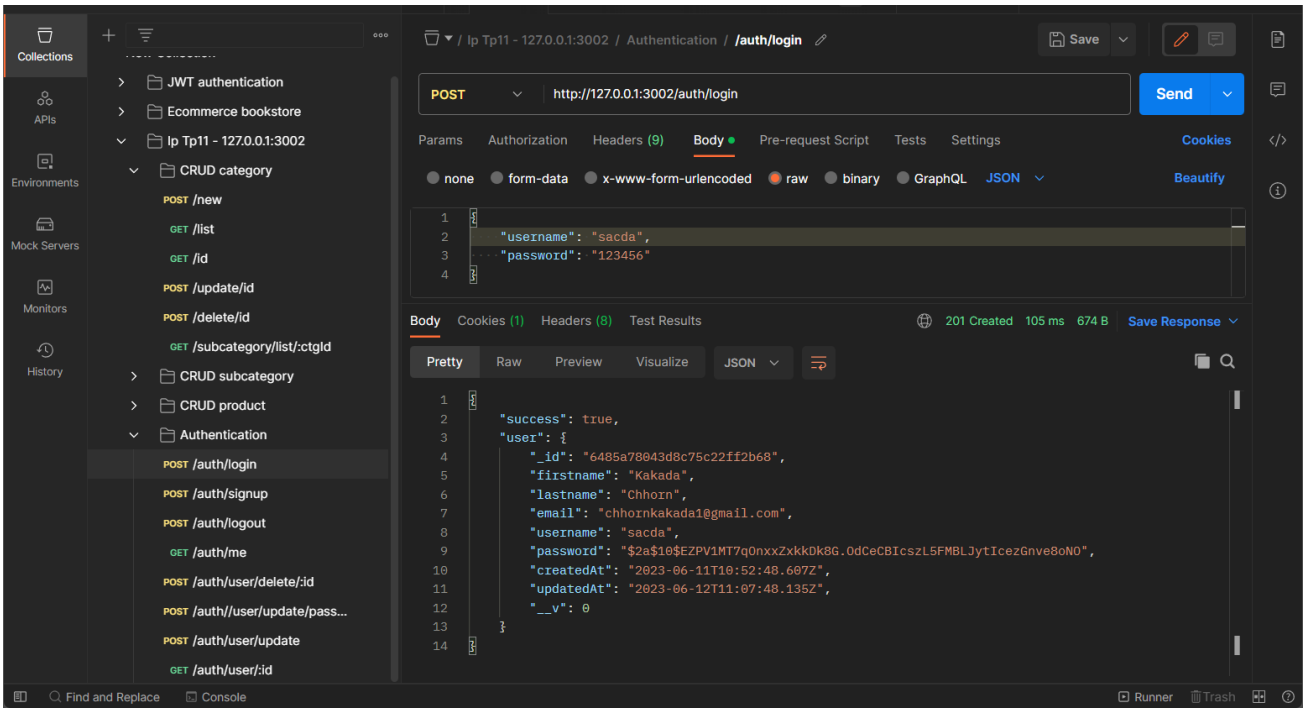
CRUD categories

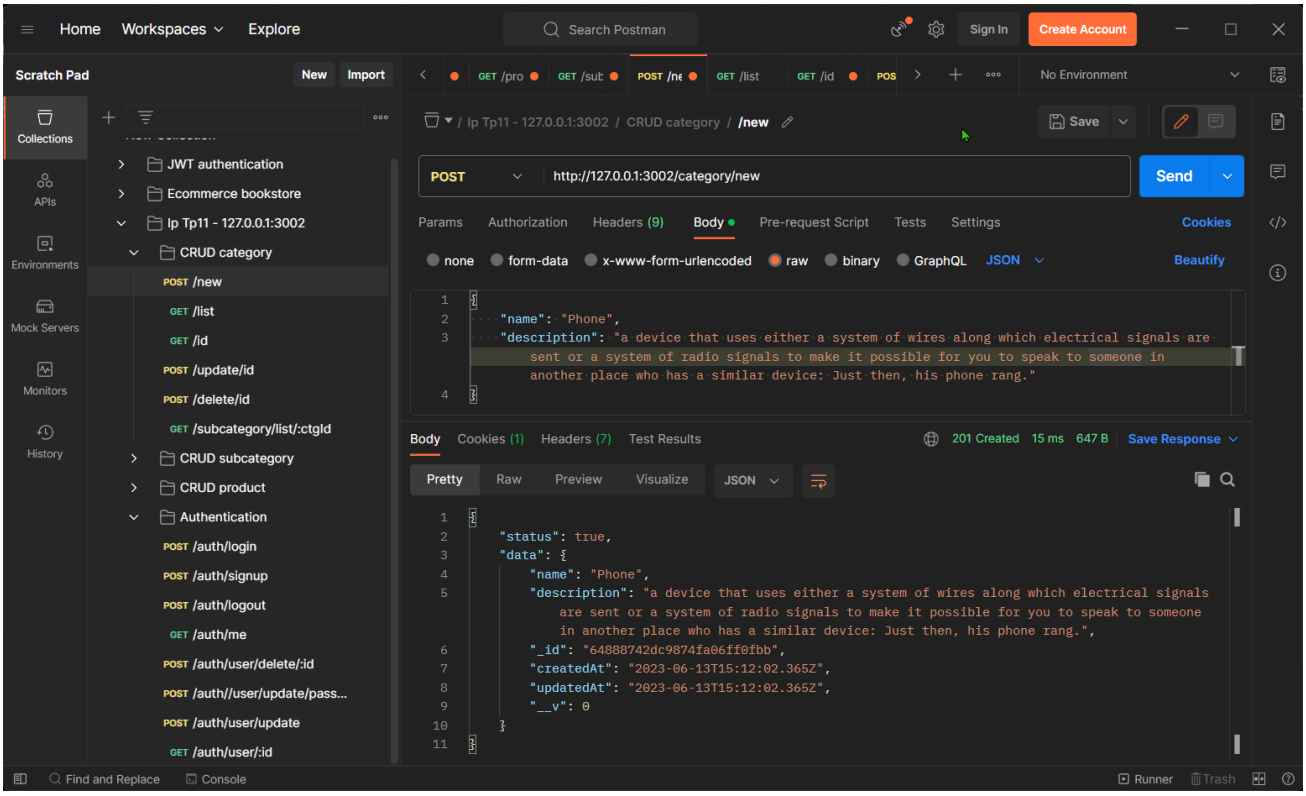
1. Add new

Cannot create new category, if the user not yet login.



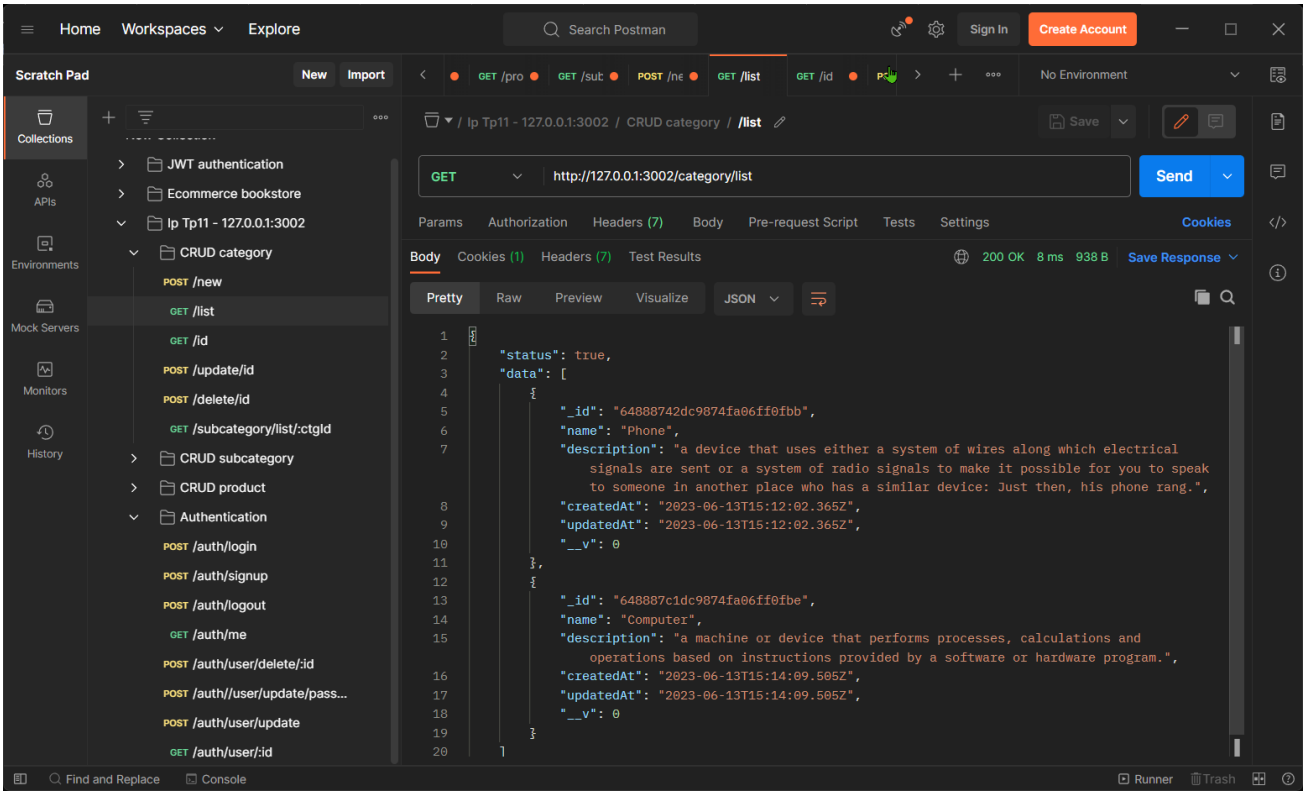
Create successfully when user already logined.





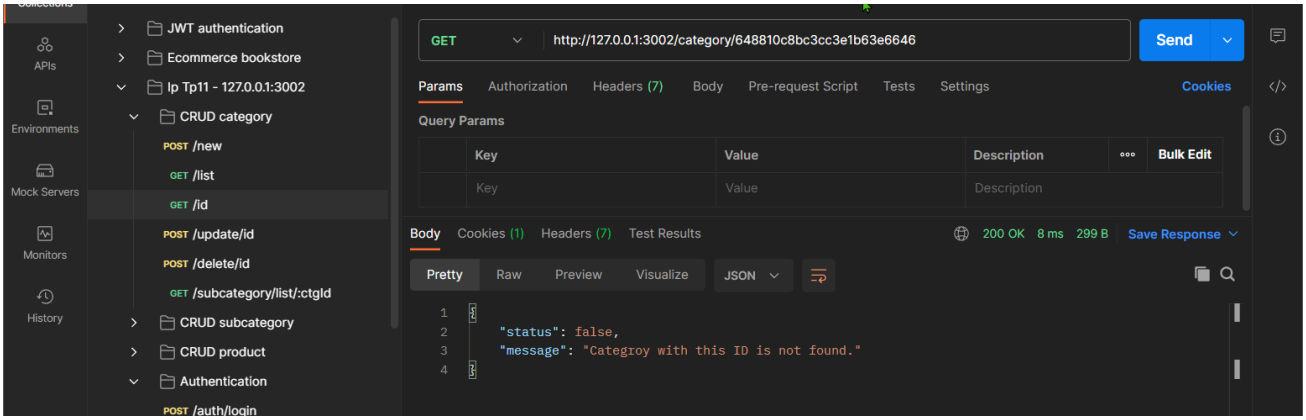
2. List all

This route list all category even if the user login or not.

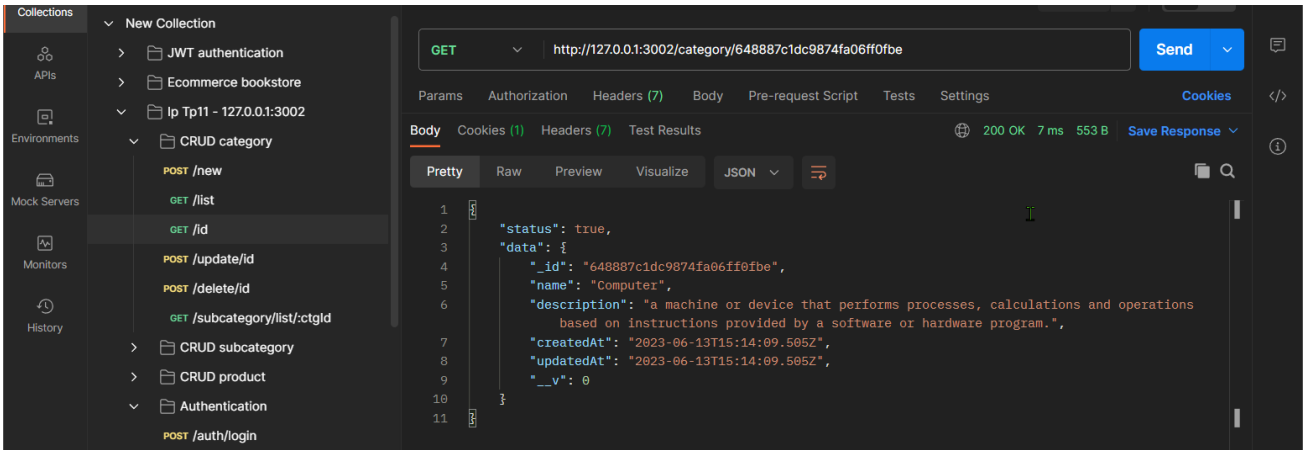


3. Find by id

When the user input wrong id.

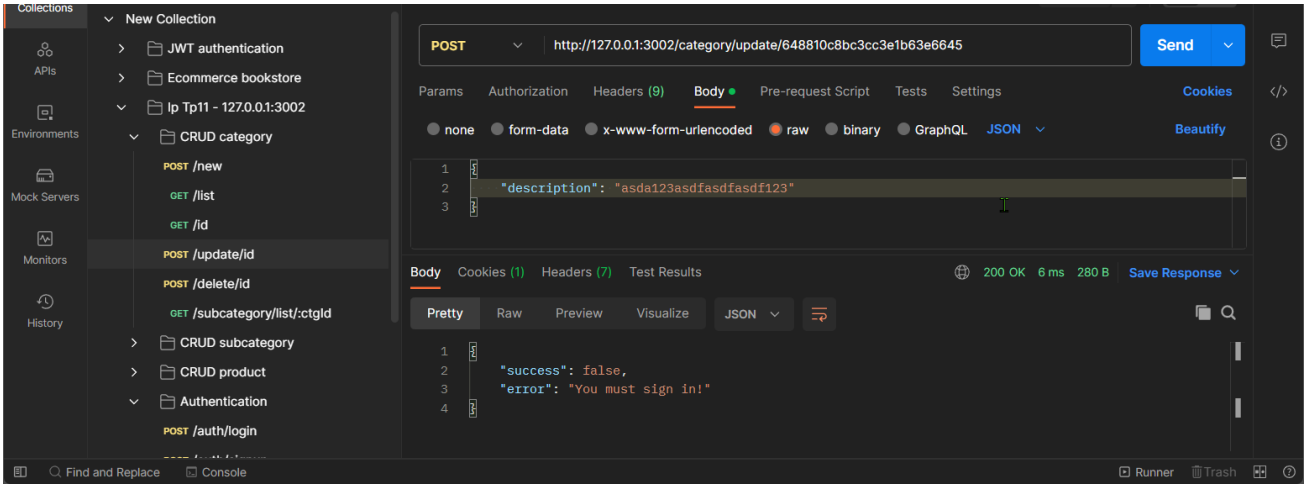


When the user input the correct id.

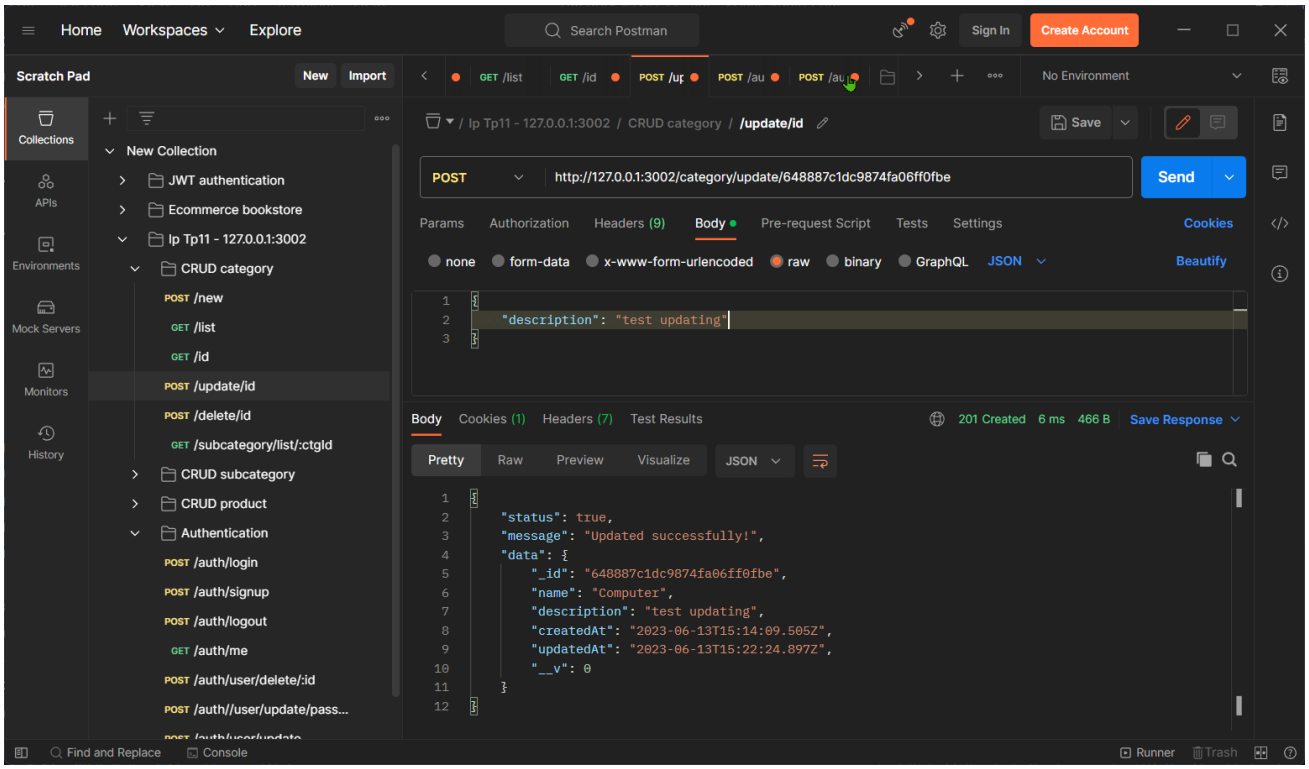


4. Update by id

The user need to login before deleting any category.

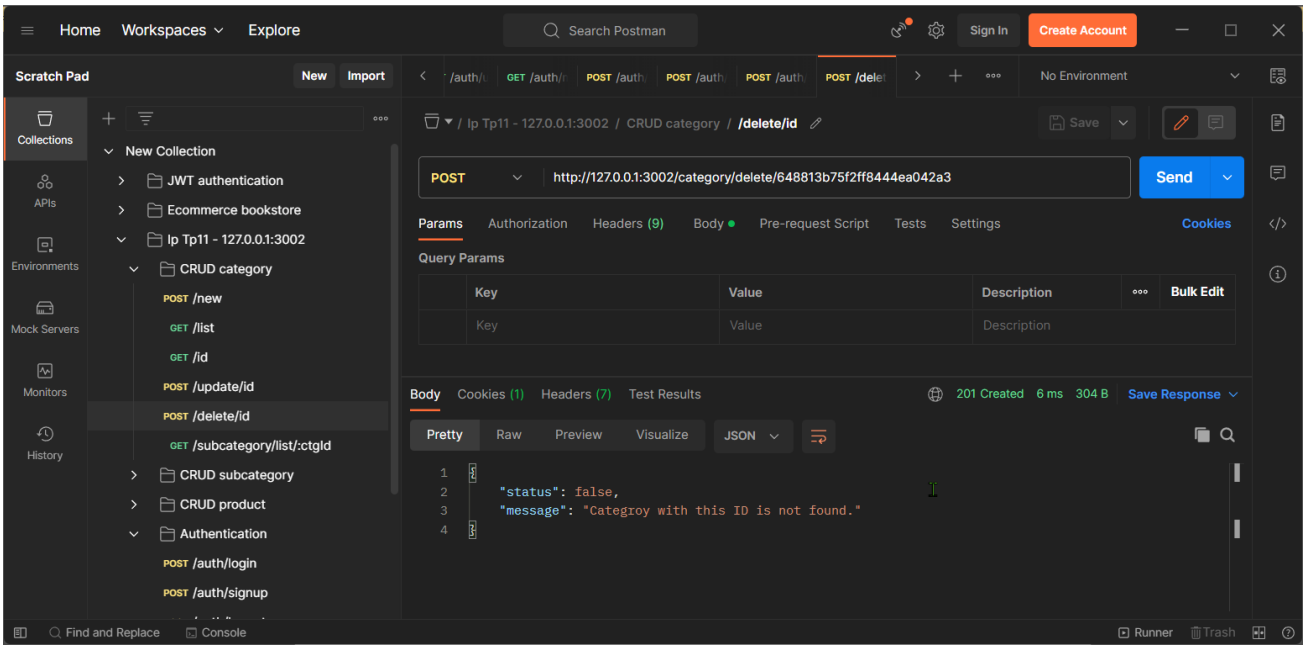


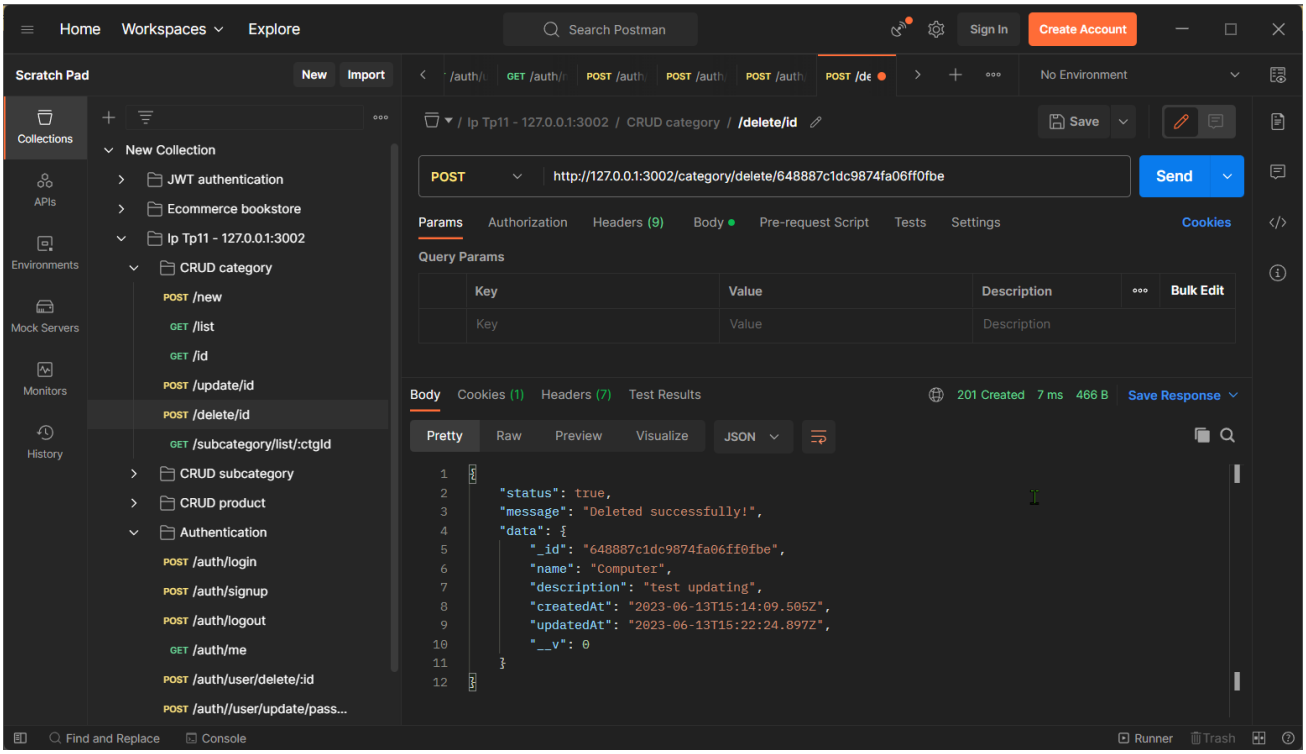
The user can update only data that they wise. Example below the user just want to update only description field of the category. But if you want to update other fields you can mention it in the body.



5. Delete by id

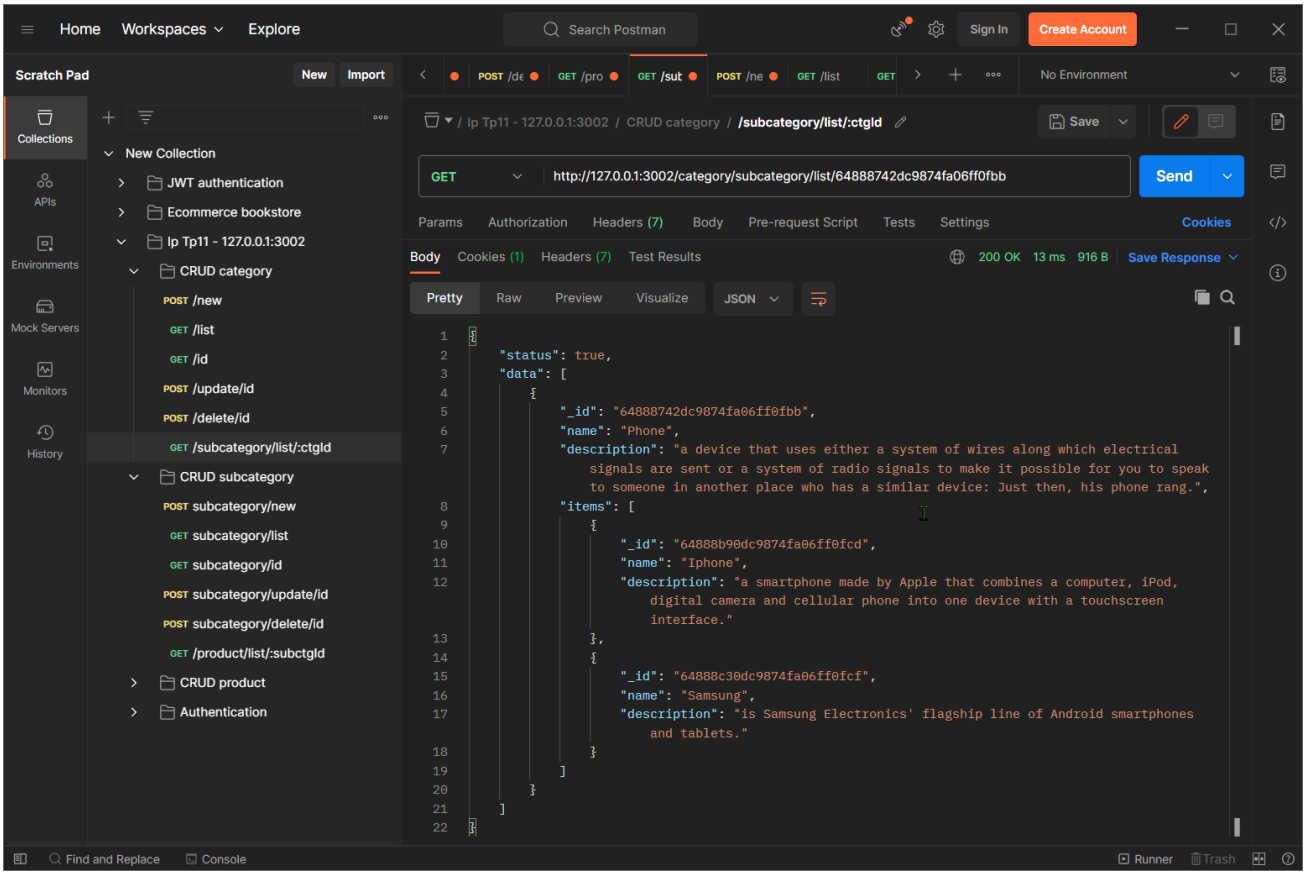
The delete route is also need to login before update too. When the user input the wrong id, it will response that the id is wrong, otherwise the category with that id will be deleted and return its category.



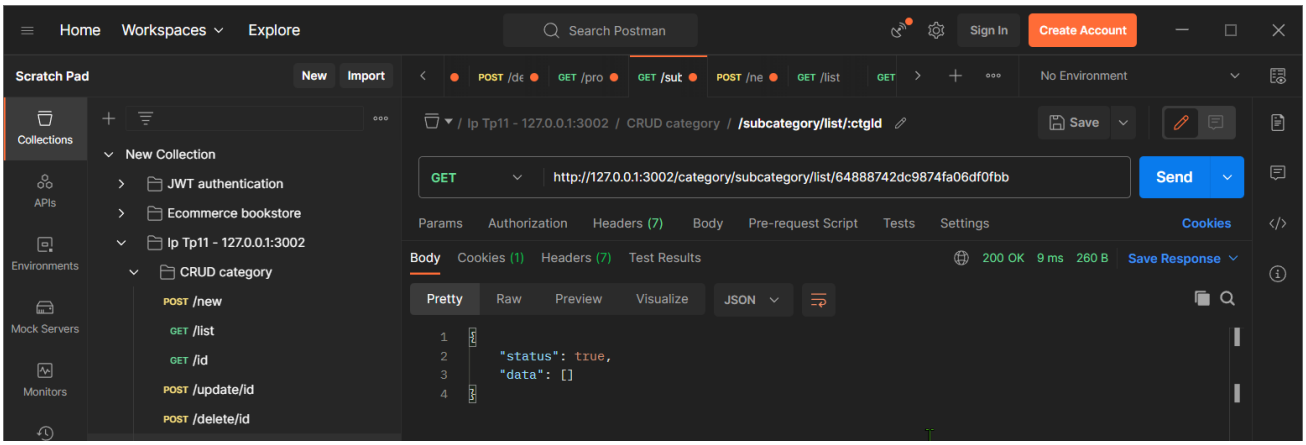


6. List all subcategories by categoryId

The output will display the list of all the category that defined by categoryID.



When the user input wrong ID of category.



7. Apply middleware in the category controllers

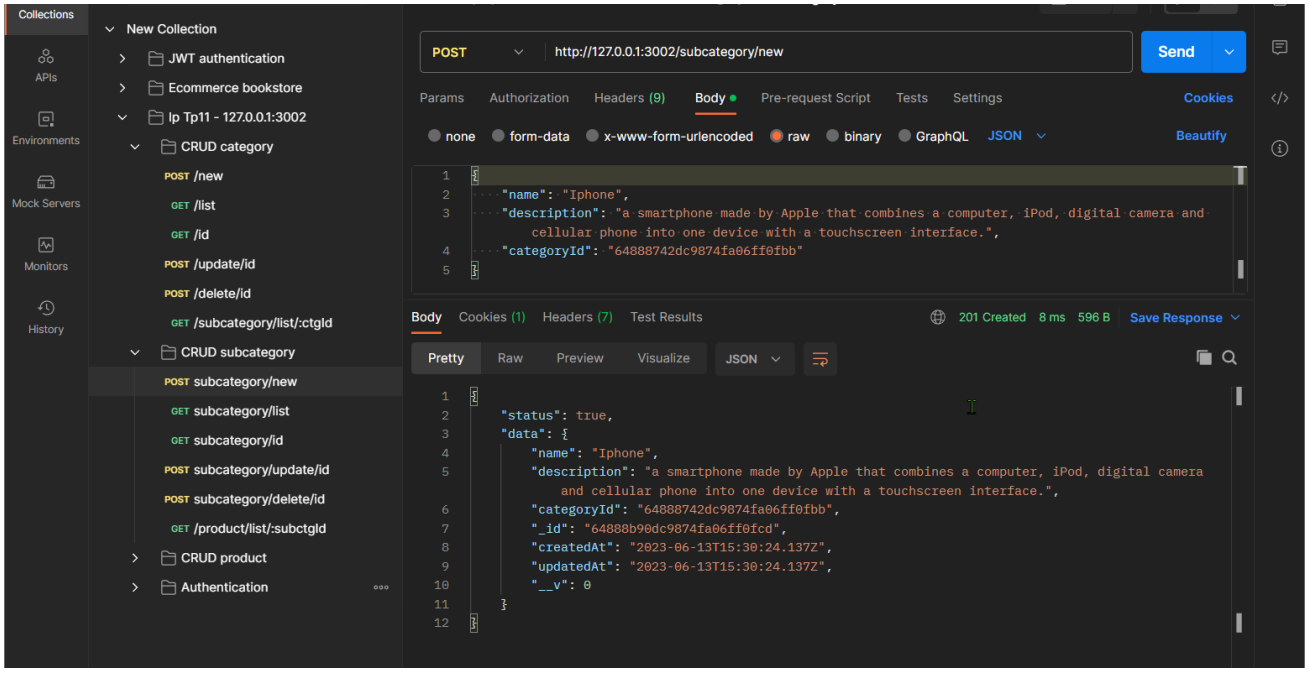
```
import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
import { CategoryController } from './category.controller';
import { CategoryService } from './category.service';
import { CategorySchema } from './schemas/category.schema';
import { MongooseModule } from '@nestjs/mongoose'
import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';

@Module({
  imports: [MongooseModule.forFeature([{ name: 'Category', schema: CategorySchema }])],
  controllers: [CategoryController],
  providers: [CategoryService]
})
export class CategoryModule implements NestModule {
  configure(consumer: MiddlewareConsumer) {
    consumer
      .apply(EnsureSignedInMiddleware)
      .forRoutes(
        { path: 'category/new', method: RequestMethod.POST },
        { path: 'category/delete/:id', method: RequestMethod.POST },
        { path: 'category/update/:id', method: RequestMethod.POST }
      );
  }
}
```

CRUD subcategories

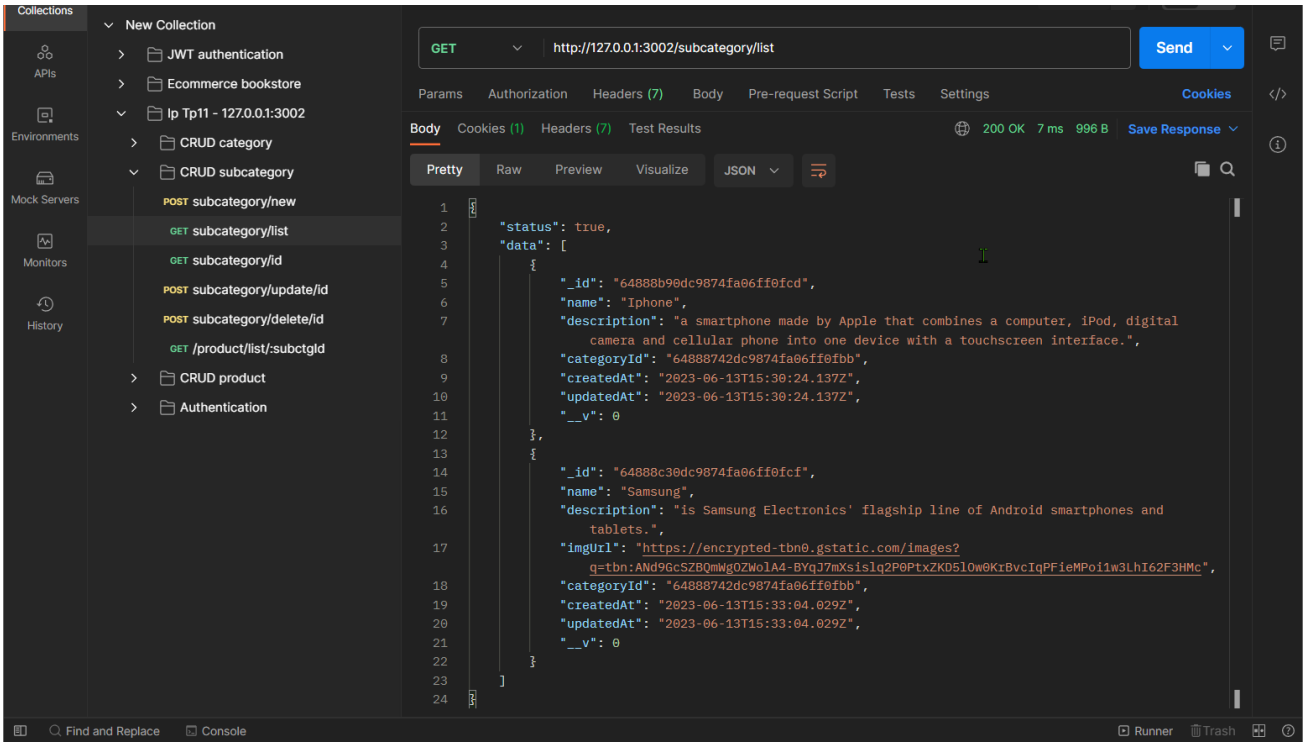
1. Add new

In this route you need to login to create a new subcategory (item).



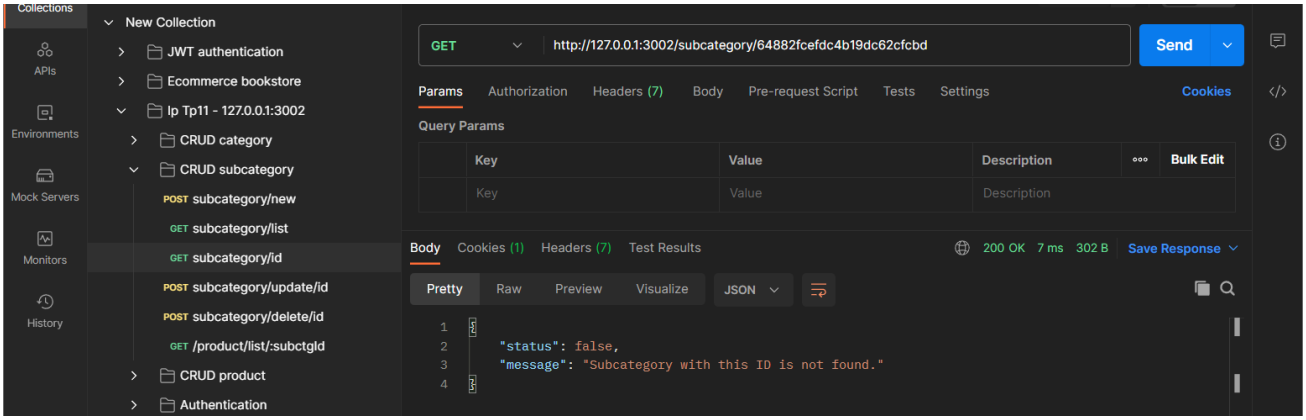
2. List all

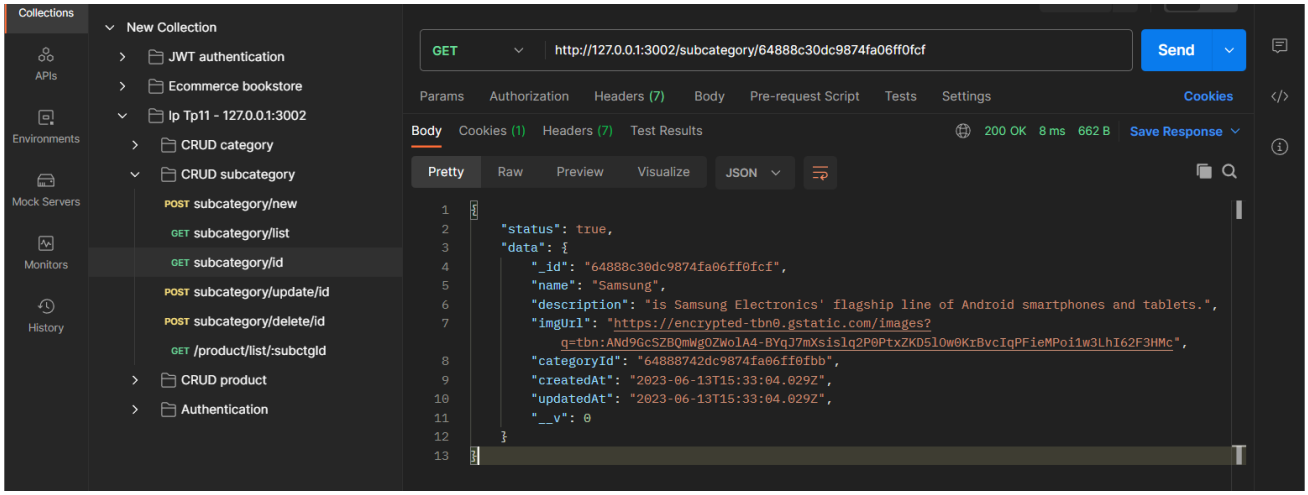
List all category even if the user is logged in or not.



3. Find by id

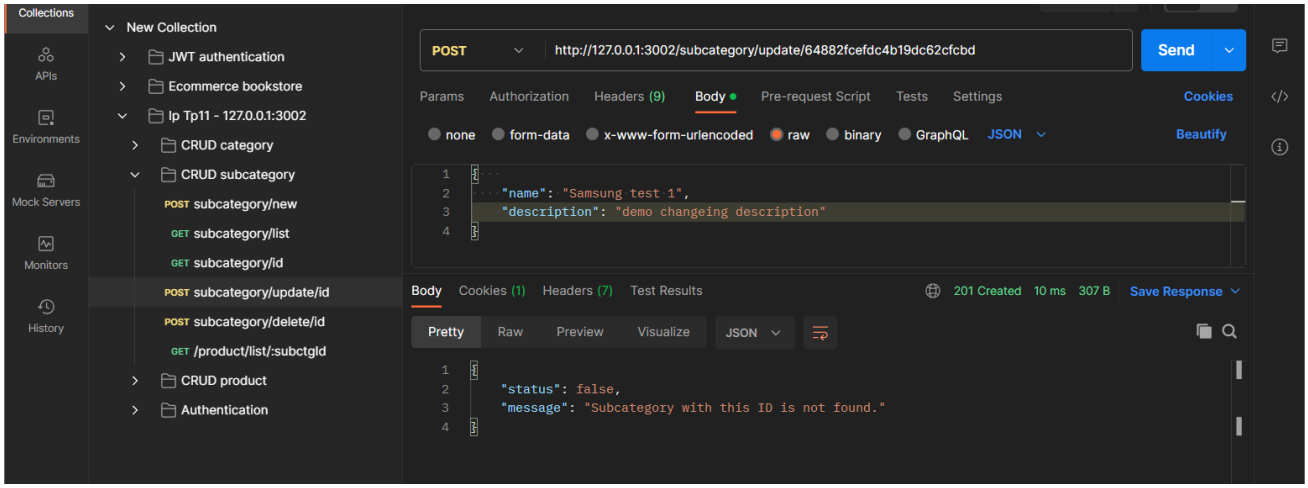
It will tell the user if the user has input the wrong ID, otherwise return the data.

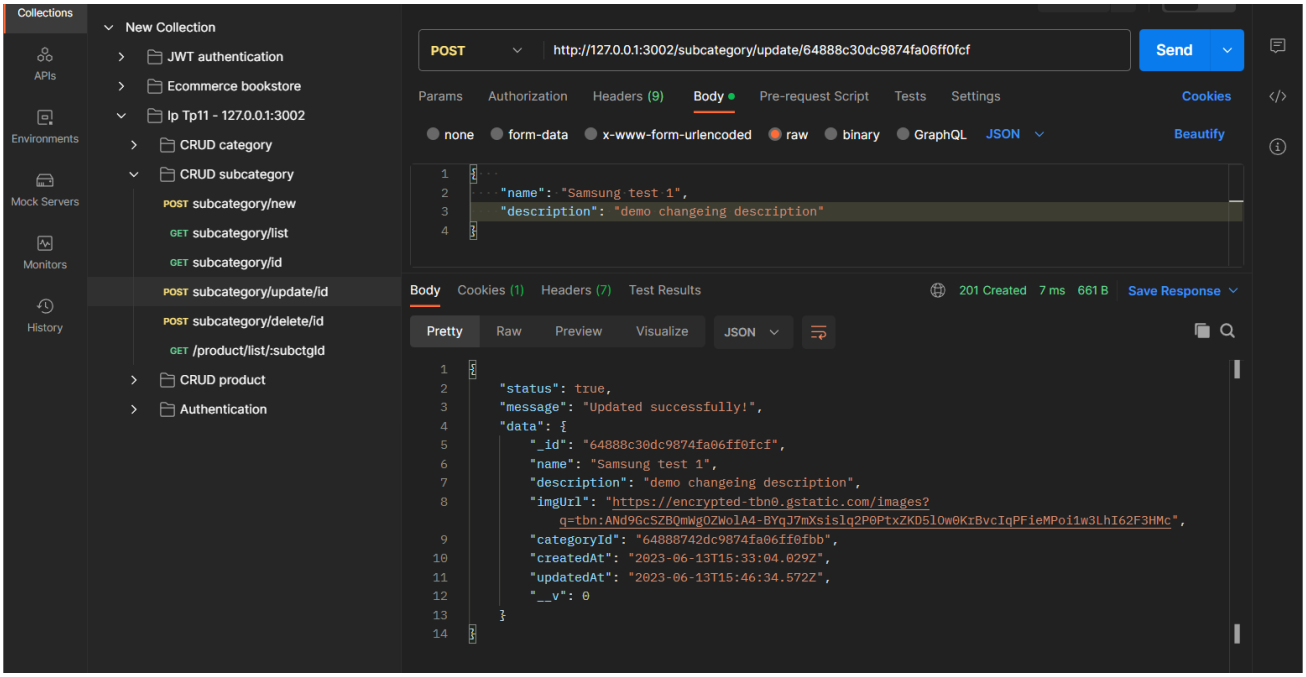




4. Update by id

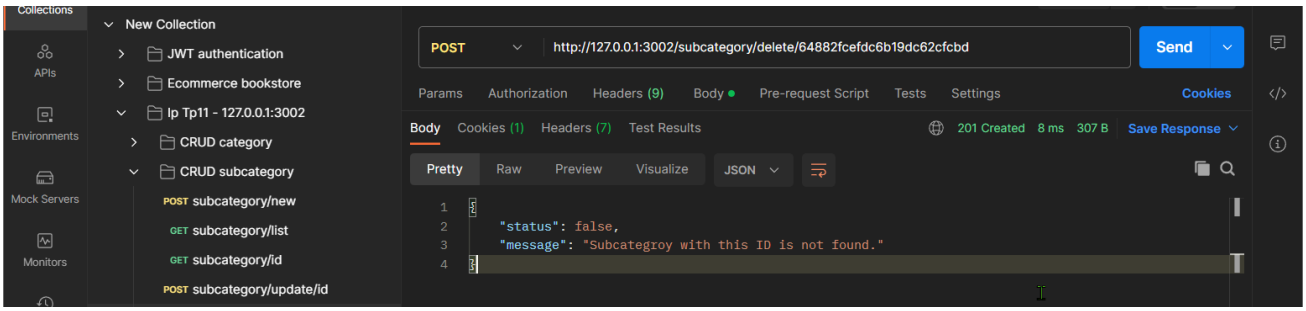
It will tell user if the user input the wrong ID, otherwise update the data.

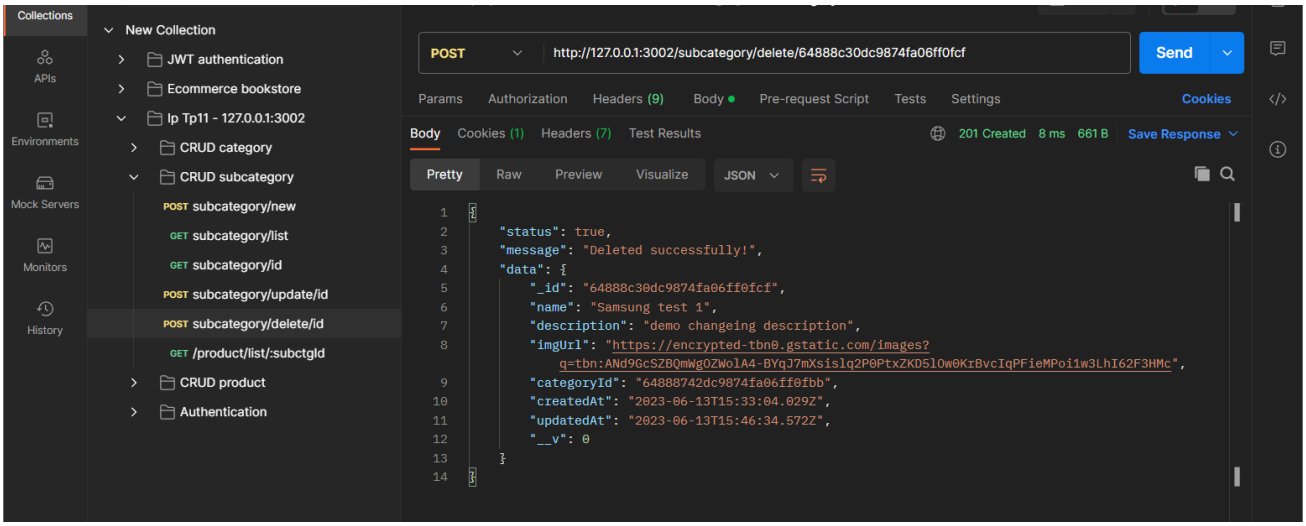




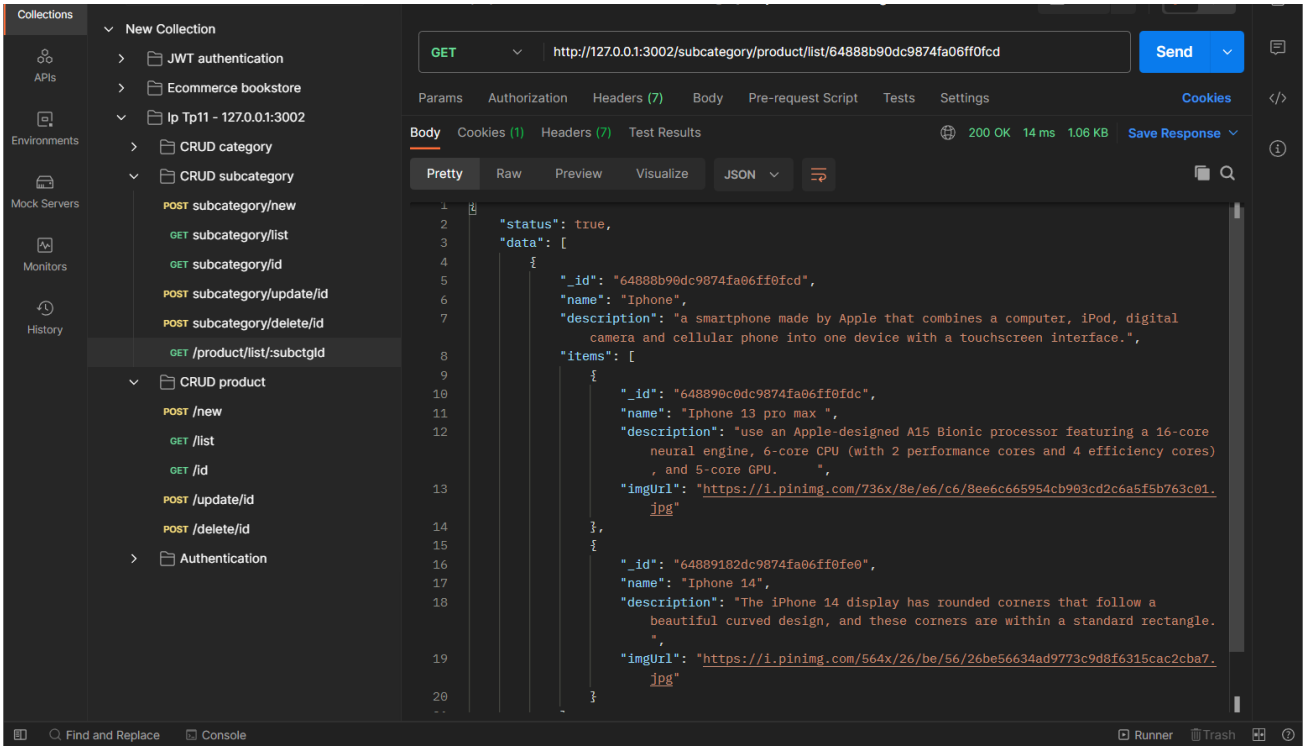
5. Delete by id

It will tel the user that user the ID is wrong when the user input the wrong ID. Otherwise delete it and return its data.





6. List all products by productId



7. Implement the middleware

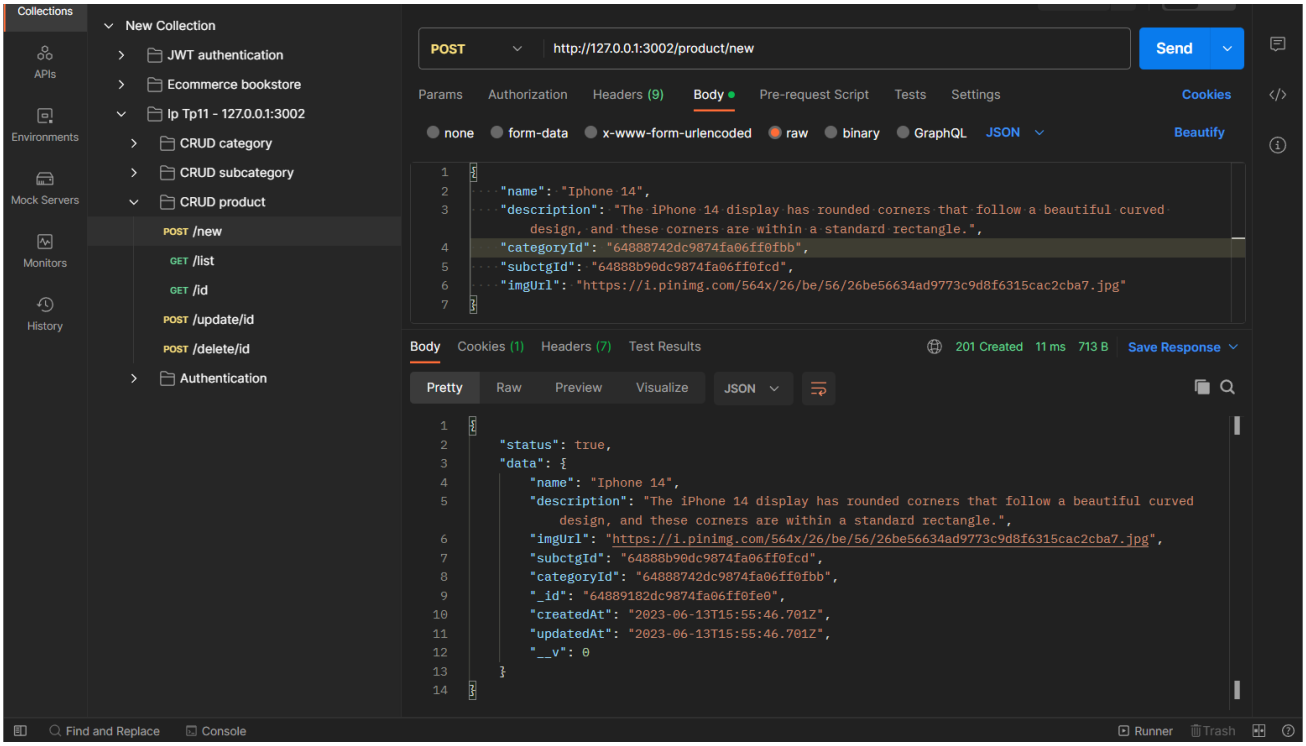
```
import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
import { SubcategoryController } from './subcategory.controller';
import { SubcategoryService } from './subcategory.service';
import { MongooseModule } from '@nestjs/mongoose';
import { SubcategorySchema } from './schemas/subcategory.schema';
import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';

@Module({
  imports: [MongooseModule.forFeature([{ name: 'Subcategory', schema: SubcategorySchema }])],
  controllers: [SubcategoryController],
  providers: [SubcategoryService]
})

// user middleware here
export class SubcategoryModule implements NestModule {
  configure(consumer: MiddlewareConsumer) {
    consumer
      .apply(EnsureSignedInMiddleware)
      .forRoutes(
        { path: "/subcategory/new", method: RequestMethod.POST },
        { path: "/subcategory/update/:id", method: RequestMethod.POST },
        { path: "/subcategory/delete/:id", method: RequestMethod.POST }
      )
  }
}
```

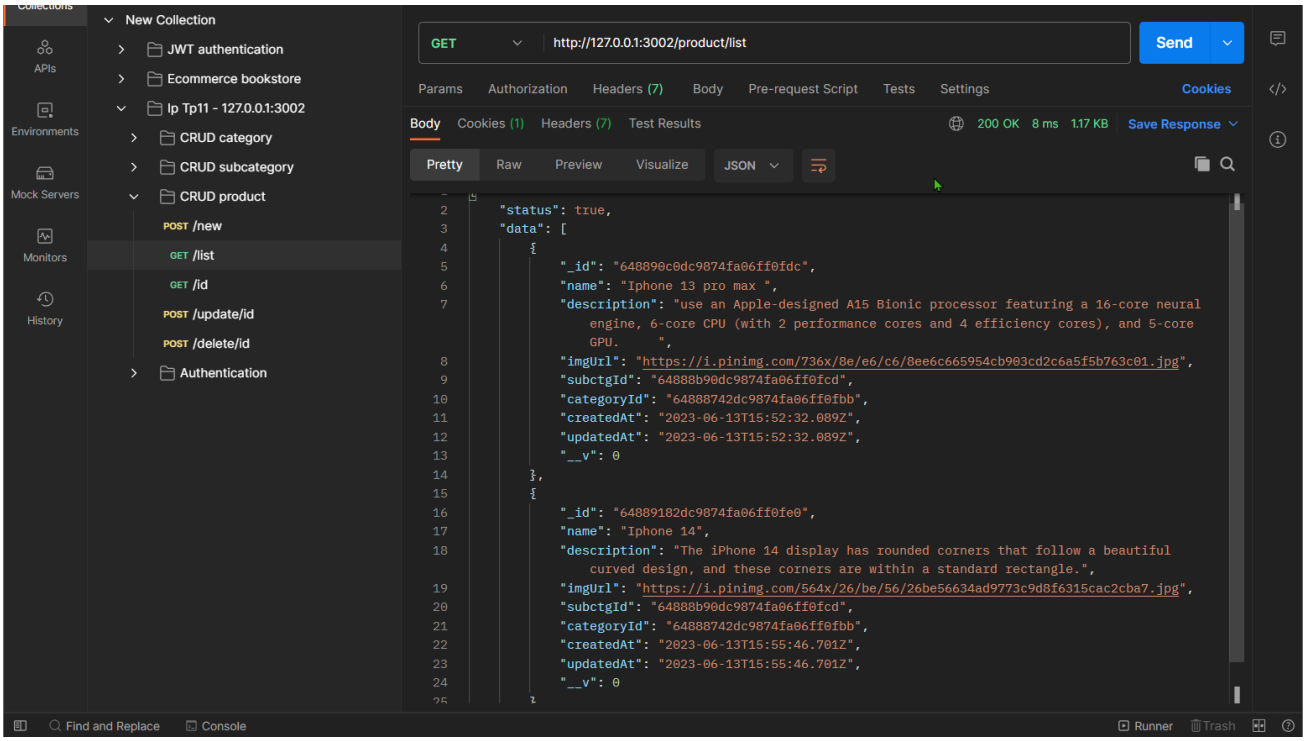
CRUD products

1. Add new



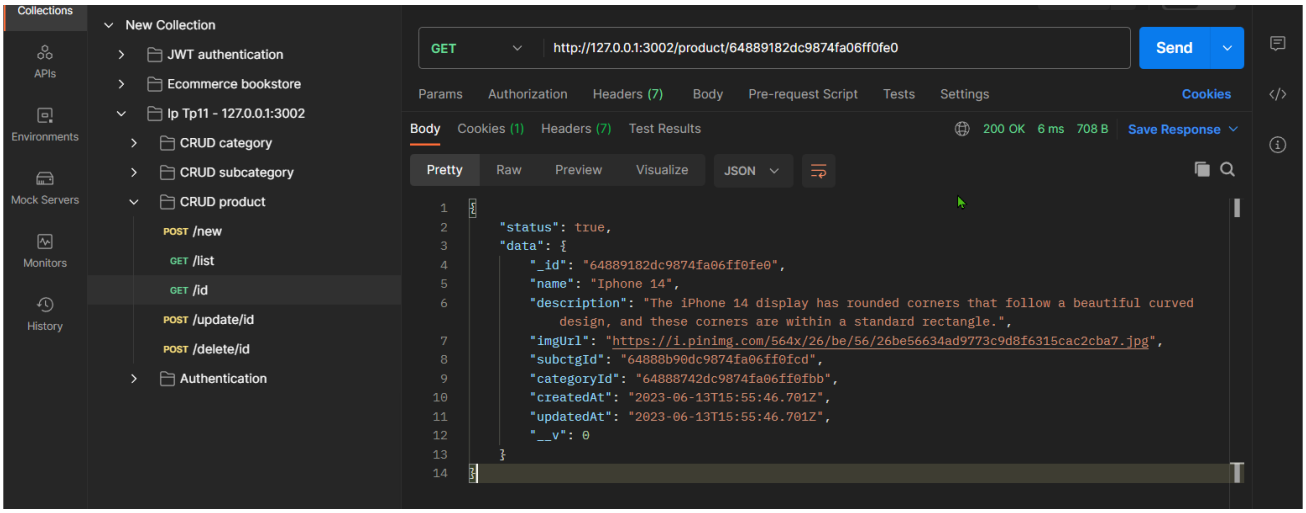
2. List all

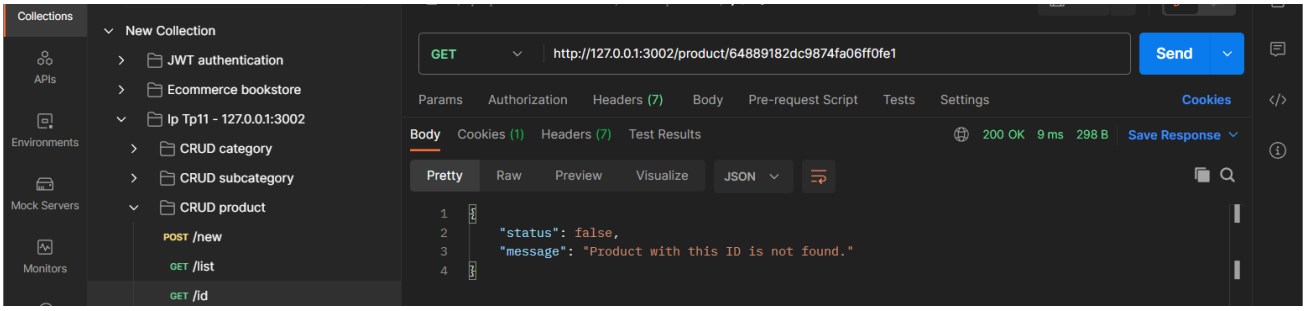
List all products.



3. Find by id

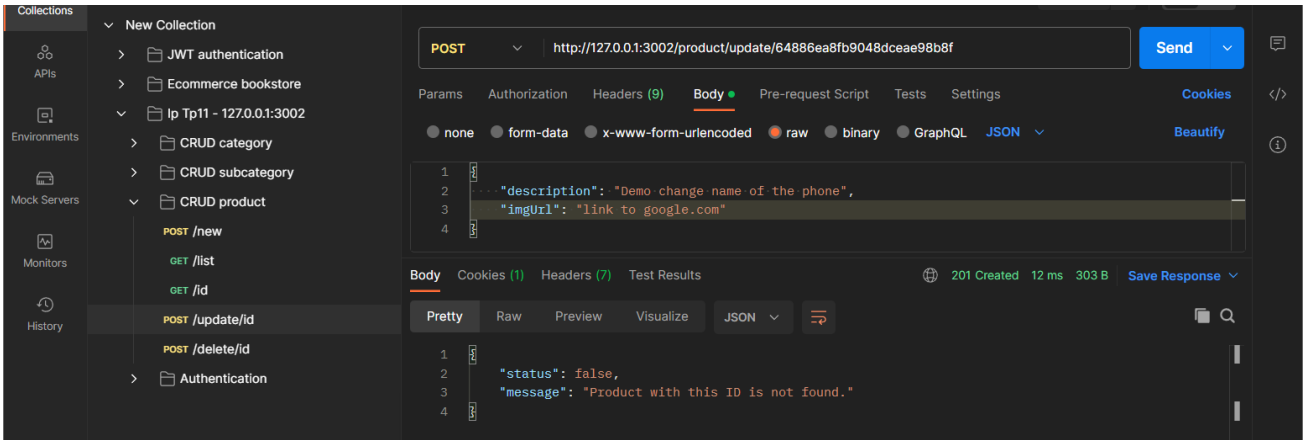
It will show the data of the product if the user input the correct ID. Otherwise, it will return the message “Product with this ID is not found”.

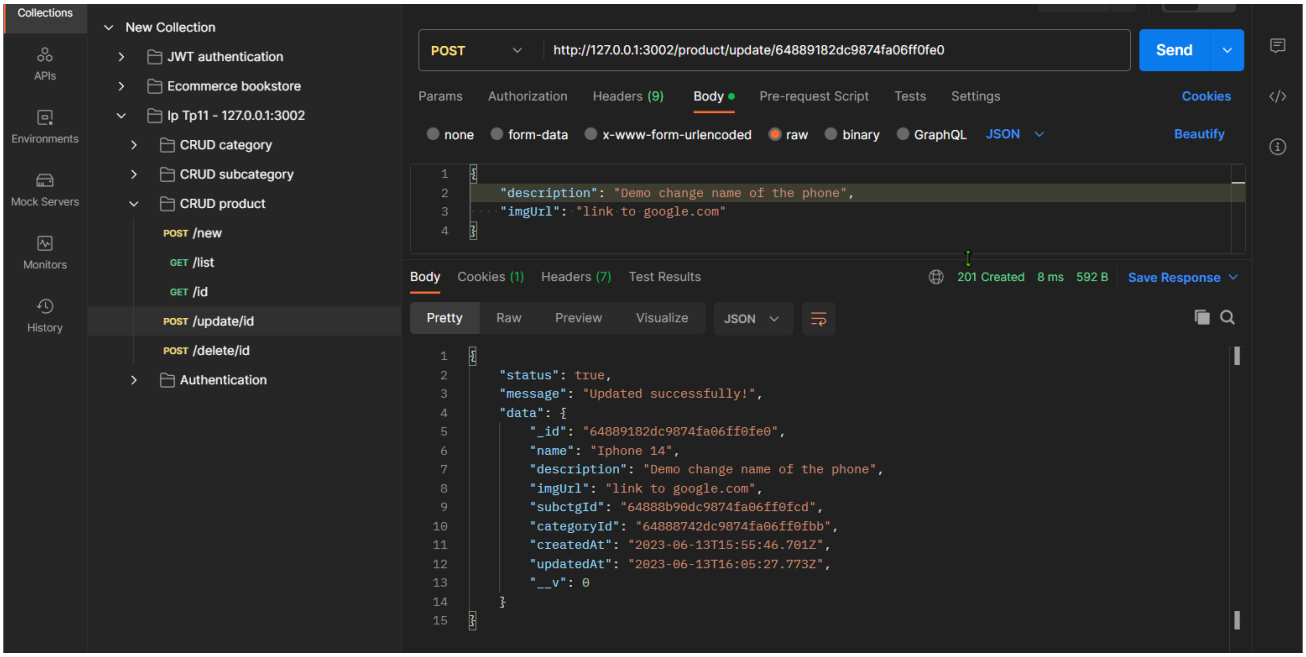




4. Update by id

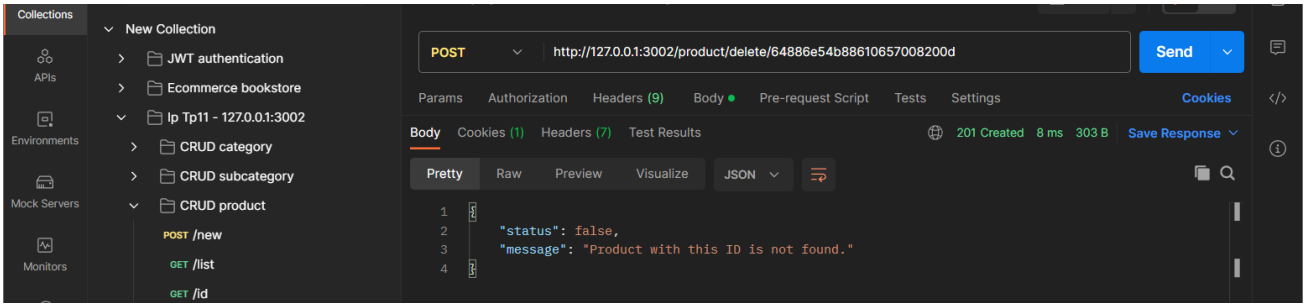
It will update if the ID is correct, otherwise it will send the message that cannot found.

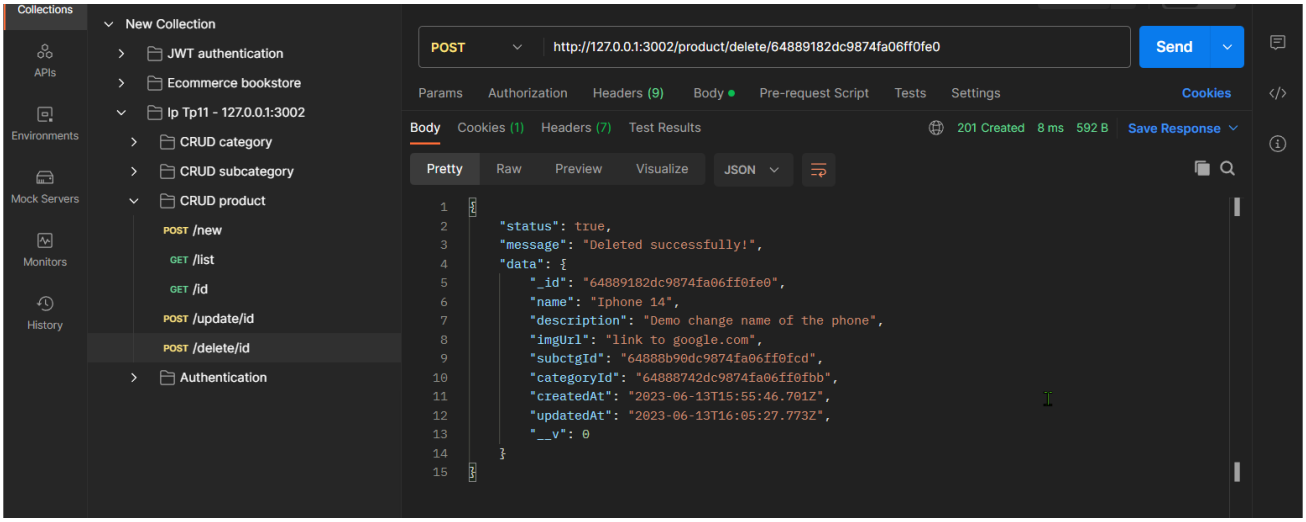




5. Delete by id

It will delete the product if the ID is correct, otherwise return with the message “Product with this ID is not found”.





6. Implement the middleware

```
import { MiddlewareConsumer, Module, NestModule, RequestMethod } from '@nestjs/common';
import { ProductController } from './product.controller';
import { ProductService } from './product.service';
import { ProductSchema } from './schemas/product.schema';
import { MongooseModule } from '@nestjs/mongoose';
import { EnsureSignedInMiddleware } from 'src/auth/middlewares/ensure-signed-in.middleware';

@Module({
  imports: [MongooseModule.forFeature([{ name: 'Product', schema: ProductSchema}])],
  controllers: [ProductController],
  providers: [ProductService]
})


// user middleware here
export class ProductModule implements NestModule {
  configure(consumer: MiddlewareConsumer) {
      consumer
        .apply(EnsureSignedInMiddleware)
        .forRoutes(
          {path: "product/new", method: RequestMethod.POST},
          {path: "product/delete/:id", method: RequestMethod.POST},
          {path: "product/update/:id", method: RequestMethod.POST}
        )
  }
}

```